FOSSpicks

Sparkling gems and new releases from the world of Free and Open Source Software

Mike Saunders has spent a decade mining the internet for free software treasures. Here's the result of his latest haul…

Programming language

nuBASIC 1.18

We're spoilt for choice with programming languages on Linux, with every paradigm under the sun represented, and returning to the clumsy spaghetti code of 80s home computers seems bonkers. So we're not advocating that people write large-scale programs in BASIC today. But nuBASIC still fills a niche: for those who fancy a trip down memory lane, for programmers who want to see how a language is implemented (the interpreter is written in C++), and for children looking for an easy path into the world of programming. You could argue that kids are better off learning Python, but the BASIC implementation here actually has elements of structured programming, and it makes it easy to handle keyboard input, graphics and so forth.

nuBASIC is provided in RPM and Deb formats, the latter of which worked perfectly on our Ubuntu 13.10 test box. Source code is also available of course – the main dependency when you're building it is SDL v2. Annoyingly, the packaged version doesn't come with a manual page, nor with any examples, so you have to grab those via **examples_1.13.tar.bz2** from the project's website. And then you might get stuck when trying to run a program; it turns out that you need to use the **-e** flag, otherwise you're dropped into an interactive session. So, run a program like so:

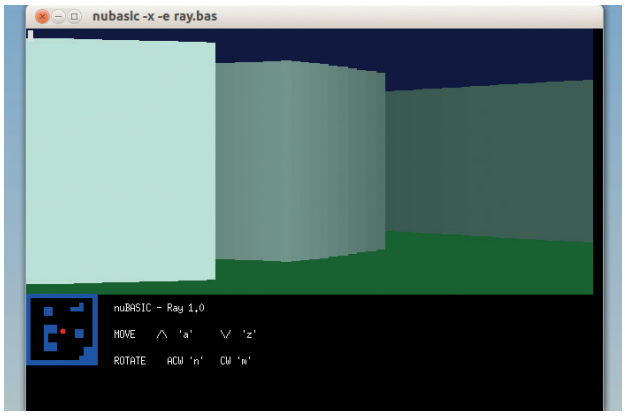
```
nubasic -e breakout3.bas
```

But! There's another hitch: the default window size is too small for many of the supplied examples, so you'll have to resize it before you use the programs.

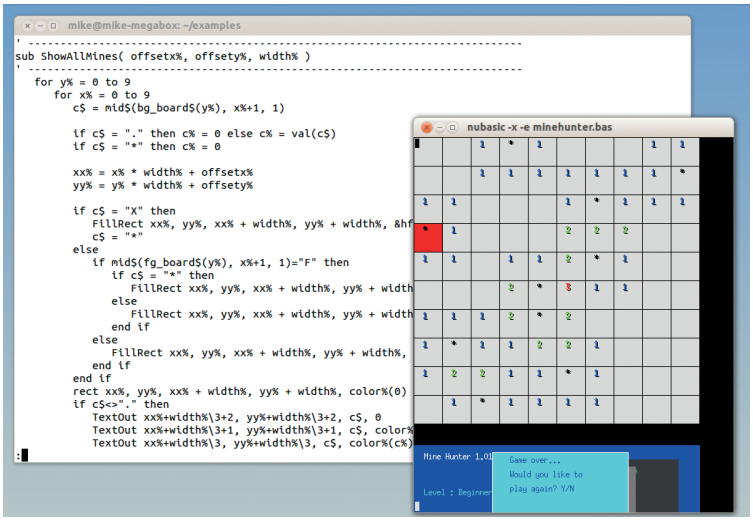
Take a look at the examples to see what the language can do. The examples bundle includes three versions of the classic block-bashing *Breakout* game; the first uses the hideous GOTO-laden line-numbered programming approach of Speccy-era machines, while **breakout2.bas** and **breakout3.bas** demonstrate the interpreter's ability to use more advanced programming methods.

Other examples include **ray.bas**, an impressive (albeit slow) 3D-esque raytracing demo, along with **minehunter.bas**, a clone of the classic *Minesweeper*. The examples show many aspects of the language, from reading keyboard and mouse input to plotting pixels and working with files. nuBASIC is well documented, with an extensive programming guide and language reference explaining the interpreter's capabilities using copious examples.

"The BASIC implementation here makes it easy to handle keyboard input, graphics and so forth."



You can write simple 3D games, as the raytracing demo shows.



Here's *Minehunter* in action, along with one of the more complicated snippets from its source code.

PROJECT WEBSITE
https://sites.google.com/site/nubasiclanguageinterpreter

Operating system

Haiku OS 2014-08-31

Linux on the desktop is a curious beast: there's no single team in charge of it all. We have the kernel hackers working in one group, X being developed by another, the Gnome and KDE coders busy elsewhere, and so forth. Distribution vendors fit it all together, and the end result is a hugely versatile desktop OS.

Now, imagine an OS created from the ground up that focuses entirely on the desktop. Unlike Linux, it doesn't have an interest in also working on big-iron mainframes or postage stamp-sized embedded devices. Everything is developed in unison – the kernel, the graphical layer, the toolkit, the desktop and the core applications. This is Haiku OS, an open source implementation of BeOS, a scorchingly fast multimedia OS that gained some small scale popularity in the late 90s (and became defunct in 2001).

It's been a while since the last alpha release, so we fired up a nightly development snapshot, which is available as a **.vmdk** virtual hard drive file, ready to use in *VirtualBox* or *VMware*. Download the Zip file, extract it and in *VirtualBox*, go to Settings > Storage and choose it as the drive image for your virtual SATA controller. (It's also available in other formats, eg for writing to a USB key – see the bottom of **http://download.haiku-os.org** for more details.)

Haiku boots impressively quickly, even inside a virtual machine, and displays a bare desktop that harks back to the days of Windows 98. There's little visual glitz here, as the Haiku team is focused on usability and performance. Click on the leaf icon in the top-right to open the main menu; this includes a number of submenus, such as Applications and Demos, where you can play around with the included software.

What's in the box?

If you're running in *VirtualBox*, networking should be enabled automatically. *WebPositive* is a *WebKit*-based browser that runs at a decent lick, while additional apps are included for accessing mail (IMAP and POP3) and playing media files. You'll even find a terminal running *Bash*, but note that this is not a Unix-like system. Switch into the **/boot/system** directory and run **ls**, for instance, and you'll see that the filesystem layout is completely different.

Haiku aims to be compatible with the last release of BeOS, although this has meant sticking with *GCC 2* until now – and that version of the compiler is 13 years old. *GCC 4* is available though, for those not interested in backward compatibility. The API is well documented, and if you're a dab hand at C++, it doesn't take much effort to knock together a quick Hello World app. Various third-party applications are available at **www.haikuware.com**, although the selection is very small when compared to the big-name distros.

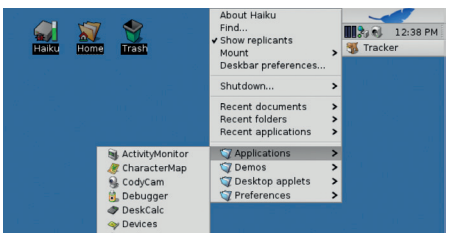
Haiku's progress has been slow in recent years, but we still cheer it on as an alternative to Linux, especially on older PCs. There's room in the market for a svelte low-latency OS with a razor-sharp focus on desktop computing – especially if it can bring new features to the table.

Haiku doesn't sport wobbly windows or fancy drop shadows, but it runs at a blistering pace.

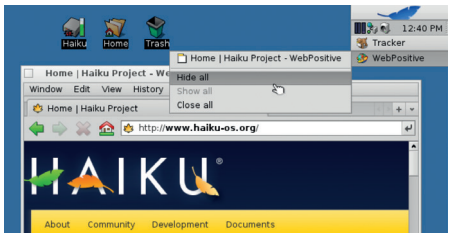
"Haiku is a svelte low-latency operating system with a razor-sharp focus on desktop computing."

PROJECT WEBSITE
www.haiku-os.org

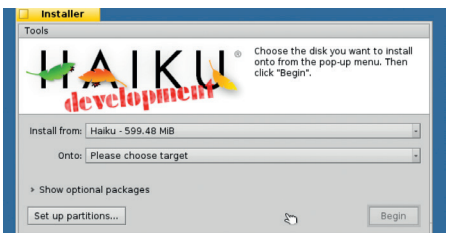
How it works: The Haiku desktop



1 Boot
Boot the hard drive image in *VirtualBox* (or the alternative image from a USB key on a real PC) and you'll arrive at the desktop. Click on the leaf icon to explore software.



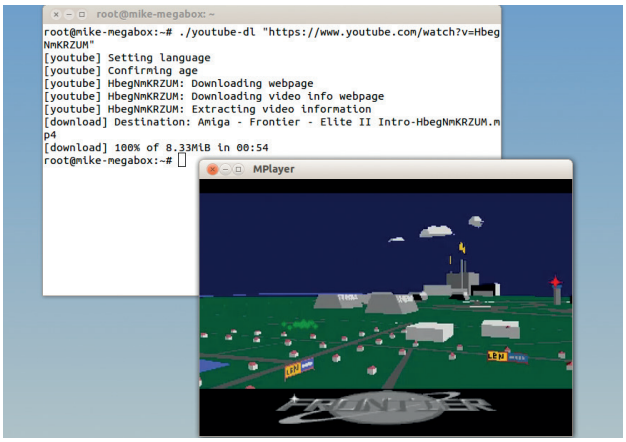
2 Run programs
When you start each program, it will be added underneath the leaf button and system tray in the top-right. This is like a taskbar – click on buttons to close apps (or use the buttons in their titlebars).



3 Install
You can perform a native hard drive installation under Applications > Installer. Note that this is still alpha software, so back up important data and don't install it on a production machine!

Video downloader

youtube-dl 2014.08.29


root@mike-megabox:~# ./youtube-dl "https://www.youtube.com/watch?v=HbegNmKRZUM"
[youtube] Setting language
[youtube] Confirming age
[youtube] HbegNmKRZUM: Downloading webpage
[youtube] HbegNmKRZUM: Downloading video info webpage
[youtube] HbegNmKRZUM: Extracting video information
[download] Destination: Amiga - Frontier - Elite II Intro-HbegNmKRZUM.mp4
[download] 100% of 8.33MiB in 00:54
root@mike-megabox:~#
MPlayer

Store videos locally (and avoid dodgy browser plugins) with this handy script.

YouTube might go down in history as the biggest time-waster ever created. Sure, there are some genuinely useful videos on there, but in all honesty we spend 99% of our time there watching cat videos and people playing games that we used to play (but can't be bothered loading up now). It's possible to download videos from YouTube, but some of the browser extensions that do this are rather dodgy, possibly sending your browsing history to unknown third parties.

We Linux users have a better solution in the form of **youtube-dl**. This is a (large!) single Python script that takes a URL and spits out a video file. It's remarkably easy to install; just grab the file from the Downloads section of the website, make it executable (eg **chmod +x youtube-dl**") and run it from your home directory like so:

```
./youtube-dl "<URL>"
```

Replace **<URL>** here with the full address of the YouTube video, as displayed in your browser. As you can see in the screenshot, **youtube-dl** grabs the page and parses it for the video content, before downloading the media. In many cases this will be a Flash (**.flv**) file – but some videos are provided in MP4 format. A decent media player like *MPlayer* or *VLC* should be able to handle both formats.

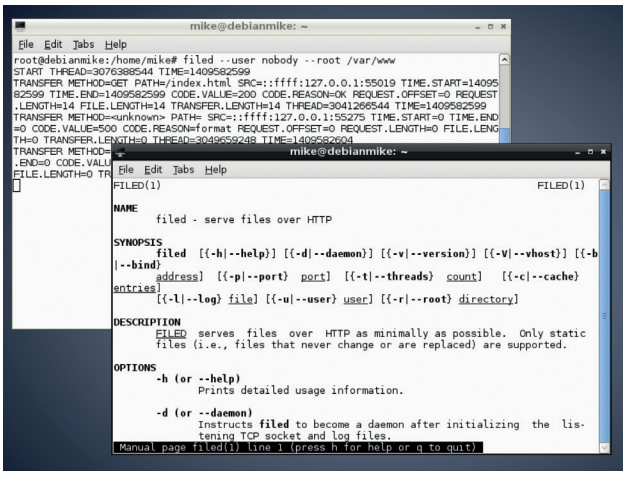
But **youtube-dl** can do a lot more: it can extract the audio from a video and convert it into a different format (providing you have the right tools installed), which is great if you've found a music video and want to keep the song on your MP3/Ogg player. You can ask it to embed subtitles into video files, log in to YouTube using a username and password, and even download adverts, if you feel guilty about not giving enough money to Google.

The program also works with video sites such as Vimeo, Vine and LiveLeak, and because these sites often change their underlying HTML (causing **youtube-dl** to break), you can always upgrade to the latest version in-place with the **-U** flag.

PROJECT WEBSITE
http://rg3.github.io/youtube-dl

Lightweight static content web server

Filed 1.8



Filed's all-caps log format (background terminal) is a bit painful on the eyes, but at least there's plenty of info.

Picture the scene: you've resurrected an old PC to see what it's still capable of. You want to share some files over your home network from it via HTTP, so you install *Apache* and... it crawls. You try another web server from the repositories, but it's equally sluggish on such limited hardware. You try yet another, and this time you end up getting bamboozled by its configuration files.

In these cases, you want the simplest, fastest, no-nonsensest HTTP server possible, and *Filed* is just that. It's a single 64k binary, with no configuration file – everything is set at the command line.

To build it, you'll need *Tcl* installed, and when you run **make** you might see an error message about a missing **mime.types** file. In this case, open the **Makefile** in a text editor and change the **MIMETYPES** line to point to **/etc/mime.types** instead of the default location. Run **make** again, followed by **sudo make install**, and you're ready to go.

By default, *Filed* should be run as the root user, and it serves up your root (**/**) directory. Obviously this isn't very useful, and potentially dangerous; to change the user (via **chroot**) and directory that's served up, run it like so:

```
filed --user nobody --root /var/www
```

Filed doesn't generate directory listings and instead attempts to serve up **index.html** by default. To boost performance, *Filed* is multithreaded with every thread serving a single concurrent client. Various extra options are available to bind to a different address or operate on another port, and instead of logging to the terminal you can redirect the output to a file.

"Filed is the simplest, fastest HTTP server possible."

PROJECT WEBSITE
http://filed.rkeene.org/fossil/index

Book writing assistant

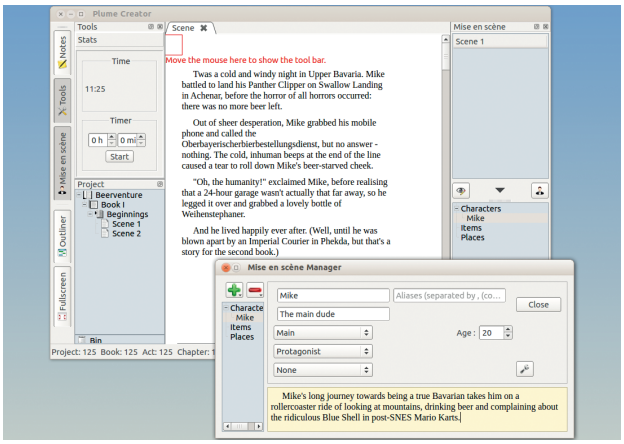
Plume Creator 0.67

They say that everyone has a book in them, but have you ever tried writing one? It's all fun and games at the start, when you're concocting plots, scenes and characters, but as the story progresses, managing everything can become a nightmare. You can try to structure things in a word processor, but a better solution is to use a dedicated novel writing tool like *Plume Creator*.

Plume's website is pretty rubbish, with little documentation on using the program. But it does show you how to install it: 32-bit and 64-bit packages for Ubuntu and Mageia are available, along with the source code. You'll need version 4 of the *Qt* libraries to install it, as the interface is built with that toolkit.

Start *Plume* and you'll be prompted to create a new project. You're asked for the type of book (eg a short novel), and you can choose how many chapters and scenes it should contain here – but don't worry if you're not sure, as you can modify them later. From here onwards, *Plume* works a lot like a regular editor, except it helps you to manage different scenes and chapters. A tree list down the left-hand side lets you quickly switch between different parts of the book, while additional tools are available such as a note-taking panel and a timer.

The mise-en-scène panel is especially useful, letting you keep track of characters, items and places. You can note here where a character was at a certain time, and what items he/she had, to avoid continuity errors. Once you're happy with your work, you can export it in a variety of formats, including ODT (as used by *LibreOffice*), HTML and plain text. There are still plenty of unfinished bits in *Plume*, but by version 1.0 it should be a great app for aspiring writers.



Plume's interface could do with some refinement, but after 10 minutes of exploring you'll get the hang of it.

"Plume helps you manage different scenes and chapters of your book."

PROJECT WEBSITE
http://www.plume-creator.eu/site/index.php/en

Convert ANSI codes to readable text

Ansifilter 1.9

Here's something interesting to try: in a terminal window, in a directory with various files and folders, enter **ls --color > list.txt**. This redirects the output of the **ls** command (with all its colour goodness) to the file **list.txt**. Now open that file in a text editor, or view it with **less list.txt**. Notice something strange? The colours aren't there – just some weird characters like:

```
ESC[01;34mfolderESC[0m
```

Ugh. What's happening here? Well, colours and effects (like bold text) are created in the terminal via ANSI codes, which involve the escape character and numbers. Any good terminal can interpret these in command output and display them properly, but when you redirect the output to a file, it just becomes plain text.

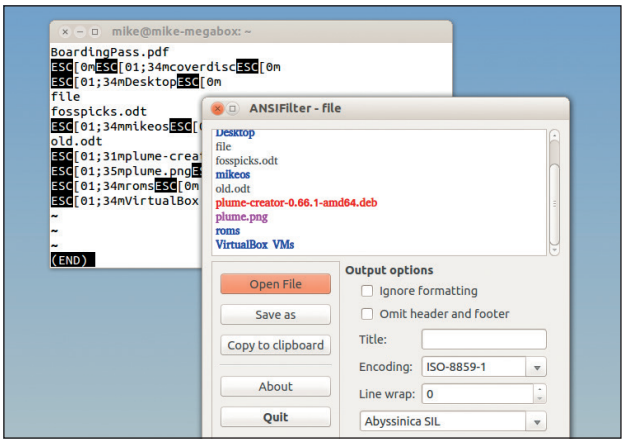
Now, say you have an important file containing these codes, eg from a log, and you want to make the information human-readable. *Ansifilter* is a godsend here: it converts the file into a better format, such as plain text, HTML, Latex, RTF or even BBCode (very useful if you want to paste the output of a command into a forum post). It's supplied as two programs, the first of which runs at the command line, and the second of which uses *Qt* to produce a simple but pleasant little GUI app.

To convert **file.log** into a HTML version called **file.html**, you'd run:

```
ansifilter -T file.log > file.html
```

Alternatively, run **ansifilter-gui file.log** to get a preview of the output, then click Save As to choose one of the formats mentioned previously. You can even change the text encoding, along with the line wrap settings and font that should be used.

Ansifilter isn't a tool you'll use on a daily basis, but it can save your life if you have a log file peppered with control codes and you desperately need to get information out of it.



ANSI codes in their raw format, and how *Ansifilter* interprets them.

PROJECT WEBSITE
www.andre-simon.de/doku/ansifilter/en/ansifilter.php

Spreadsheet app

mtCellEdit 2.4

The flagship spreadsheet program for Linux and other FOSSy systems is *LibreOffice Calc*. We already have a lighter alternative in the form of *Gnumeric*, which is darn good by the way – but *mtCellEdit* is even smaller. It's a very basic spreadsheet program, lacking many of the features and frills you'll find in the bigger tools, but for basic calculation jobs it's great.

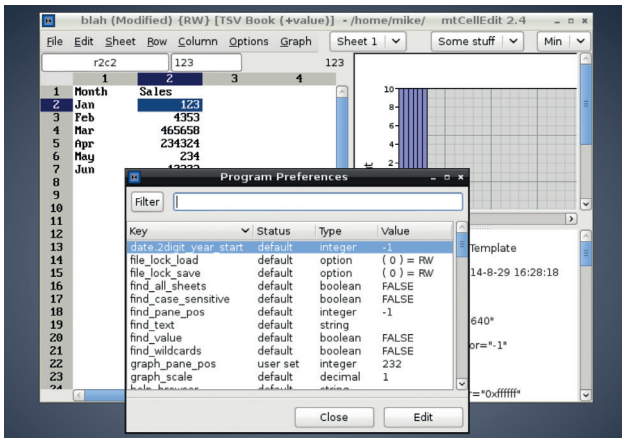
What's not so great, though, is the compilation process. When you extract the tarball you're faced with a bunch of directories containing different parts of the program – and a **README.txt** that doesn't provide much info. It does, however, point you at the project's HTML documentation, which explains the order in which you need to compile the components. The *GTK 2* toolkit is the main dependency.

Start the program and you'll see that *mtCellEdit*'s interface is as bare as they come: you have a grid for entering data, a drop-down list to switch between different sheets, and a handy list in the top-right showing values for selected cells (eg sum, maximum, average).

mtCellEdit refers to individual cells by their row and column numbers, so if you want to display the sum of columns 1 and 2 in row 1, you'll use this command:

```
=r1c1+r1c2
```

That's rather different to the A1, B2 etc system used by other spreadsheets, and takes a while to get used to. It's possible to generate bar charts in the program, although we found this cumbersome, requiring copying and pasting chunks of data into a text file, and having to do a lot of manual fiddling to get it right. *mtCellEdit* can open and save CSV and TSV (comma and tab separated value) files, though, so it's easy to share data with other apps.



There are plenty of options to tweak, but they're not presented in the most human-friendly fashion.

"For basic calculation jobs, mtCellEdit is great."

PROJECT WEBSITE
http://code.google.com/p/mtcelledit/

Scripting language

PHP 5.6.0

PHP gets a lot of flak from many developers; they regard it as a toy language that has become ugly and bloated over the years, lacking logical design and consistency. Even Rasmus Lerdorf, the creator of PHP, said that he had "absolutely no idea how to write a programming language" at the beginning. On the other hand, it's useful for cooking up quick websites on a LAMP stack, and many well-known web apps such as *WordPress* are built with it.

Anyway: PHP 5.6.0 was released at the end of August, and it brings a bunch of improvements, many of which have been in discussion for a while. High up on the list is support for constant scalar expressions, where you can use expressions in which PHP previously expected static values. For instance, you can now do this:

```
const ONE = 1;
const TWO = ONE * 2;
```

You can use them in other places like default function arguments too – the idea is to make code easier to read and more expressive. Then there's better handling of variable length argument lists for functions, so instead of messing around with **func_num_args()** and the like, you can start a function like so:

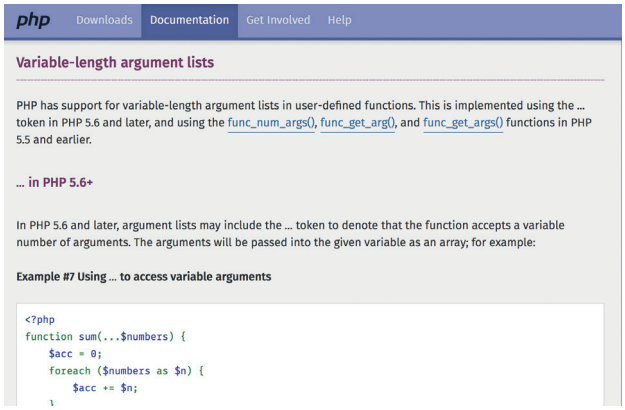
```
function sum(...$numbers) {
```

Thanks to the **...** token, this places all the arguments into an array called **$numbers**, over which you can iterate using **foreach**.

Exponation using the ****** operator is now supported in PHP 5.6.0, which means you can do this:

```
$c = $a ** $b;
```

Where **$c** contains the result of raising **$a** to the **$b**'th power.


php
Downloads
Documentation
Get Involved
Help
Variable-length argument lists
PHP has support for variable-length argument lists in user-defined functions. This is implemented using the ... token in PHP 5.6 and later, and using the func_num_args(), func_get_arg(), and func_get_args() functions in PHP 5.5 and earlier.
... in PHP 5.6+
In PHP 5.6 and later, argument lists may include the ... token to denote that the function accepts a variable number of arguments. The arguments will be passed into the given variable as an array; for example:
Example #7 Using ... to access variable arguments
<?php
function sum(...$numbers) {
$acc = 0;
foreach ($numbers as $n) {
$acc += $n;

As usual, PHP's new features are well documented, with examples showing how you can incorporate them into your own code.

Many other improvements and tweaks have been made around the codebase too: the **phpdbg** debugger has been integrated into the core function and constant importing is now possible with the **use** keyword; and file uploads of larger than 2GB are now supported. This release might not win over all the naysayers, but it's a solid job.

PROJECT WEBSITE
www.php.net

FOSSPICKS Brain Relaxers

Space trading/combat game

Oolite 1.80

We at Linux Voice HQ all have misspent youths thanks to David Braben and Ian Bell. While other kids were being cool, playing sports and chasing girls, we were perfecting docking sequences and selling robots on the black market in Sol. Yes, we loved *Elite* (and its sequel *Frontier*), and as *Elite: Dangerous* is getting tantalisingly close to release, we've been playing some open source *Elite*-ish games too.

Oolite is the arguably the best, and recently received a major update, bringing it to version 1.80. You can grab it in 32-bit or 64-bit versions from the game's website – we did the latter, and installed it like so:

```
tar xfv oolite-1.80.linux-x86_64.tgz
./oolite-1.80.linux-x86_64.run
```

We asked for the game to be installed in our home directory, and a menu icon was created under Games. (The installer also tells you how to run it manually.)

There are three main modes to *Oolite*: Normal is the full game, taking the core gameplay of *Elite* and adding lots of extra goodies. There's a tutorial mode for new players, along with a Strict mode, which aims to ape the original as closely as possible.

Version 1.80 brings about more variety in the galaxy maps, and more combinations of non-player characters, such as packs of pirates working together. You, as the player, now have a reputation, so if you're a skilled bounty hunter then many pirates will stay out of your way. It's also now easier to install expansion packs – a darn good thing, given that there are over 500 of them...

The HUD is almost identical to *Elite*'s, but the planets and spacecraft look a jillion times better.

PROJECT WEBSITE
www.oolite.org

Board/puzzle game

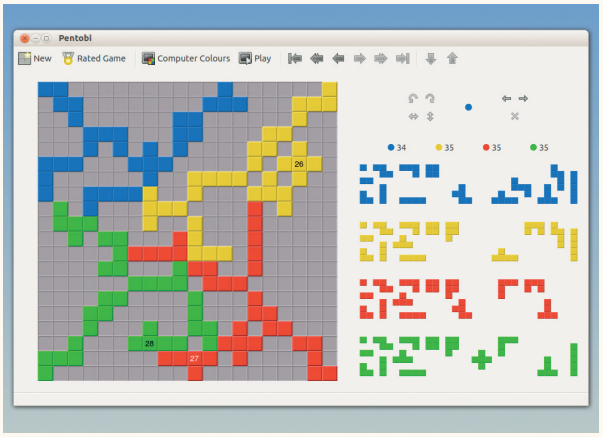
Pentobi 8.1

Looking at the screenshot, you might be tempted to think that *Pentobi* is yet another *Tetris* rip-off, and fair enough - the formula has been done to death. But although *Pentobi* uses similar shaped pieces, it's a very different game. For starters, it's based on a board game called *Blokus* that was invented in 2000, and it's great fun in multi-player mode.

Pentobi's main dependency is *Qt*; usefully, it can be built with version 4.x or 5.x of the toolkit. When you start the game, you're presented with a blank board, and by default it's you vs three CPU-controlled opponents. (Click on the Computer Colours button at the top of the window to replace CPU players with real-life human ones.)

The rules are like so: each colour takes it in turns to place a piece on the board, starting with the blue player. On the right-hand side is a palette of pieces from which you can choose – ranging from single-block pieces to five-block ones – and you can only use each piece once. You place your first piece in your designated starting corner, and subsequent pieces have to touch the same colour on the corners, but not directly on the edges. So you end up building a construction out of your corner.

However – as the other players build their constructions, there's less and less space on the board. You have to plan ahead to place as many of your pieces as possible. The game ends when nobody can place anything else, and a score is totalled based on how many pieces you didn't place. It's challenging, addictive, and gets the brain ticking over...



It's early days, but blue is getting trapped here, thanks to sneaky CPU opponents...

PROJECT WEBSITE
http://pentobi.sourceforge.net